Ruby on Rails

Charles Severance

Textbook: Build Your own Ruby on Rails Application by Patrick Lenz (ISBN:978-0-975-8419-5-2)

What is Ruby?

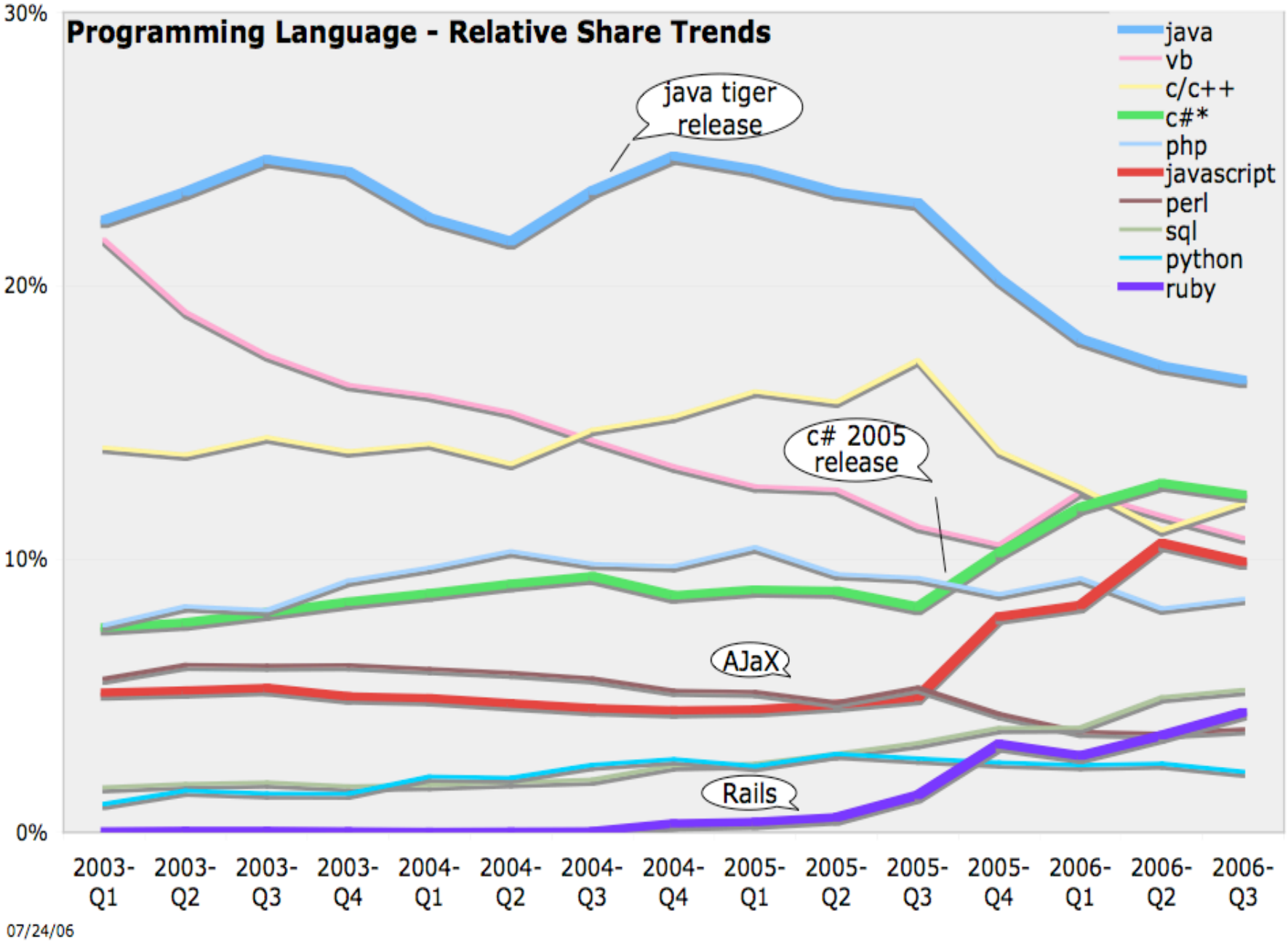
- Ruby is a programming language
  - At one level any programming language can be used to solve any problem - everything is *possible* in any language
  - Languages differentiate them selves based of ease of use, elegance, power, simplicity, efficiency, and many other subtle but very important factors

Ruby's Advantages

- Truly Object Oriented from the ground up - very consistent patterns to the language
- User-friendly language syntax - easy to read and easy to learn
- Help the user develop nice, clean, powerful programs
- Very powerful run time class extensions - make applications like Rails not only possible but very natural

Ruby History

- Originally developed in Japan by Yukihiro Matsumoto (aka “Matz”)
  - “I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language.”
- Work started in 1993, and released it to the public in 1995.
- "Ruby" was named as a gemstone alluding to name of Perl
- Also pearl is the birthstone for June, and ruby is the birthstone for July.
- English version in 1998



http://radar.oreilly.com/archives/2006/08/programming_language_trends_1.html

What is Rails?

- Rails is a full-stack web application framework
  - The Rails framework includes common functionality that is useful across a wide range of web applications
  - By starting with Rails - you get a lot for “free” - you can focus on your application and its data rather than writing lots of plumbing
- Rails is written in Ruby

Rails History

- David Heinemeir Hansson from Denmark developed a web application called Basecamp
- Rails was created by taking the general purpose bits of the Basecamp application out so they could stand on their own
- Rails 1.0 was released in December of 2005.

http://www.youtube.com/watch?v=mp4z2eK1Avw

Rails Development Principles

- Convention over Configuration
- Don’t Repeat Yourself
- Agile Development
- Scaffolding

- Model - View - Controller Architecture

Convention Over Configuration

- Add a column to a database table and in-memory objects immediately have new methods supporting the column - no need to change configuration many places throughout the application
- A reaction to Java + Spring + Hibernate which has lots of flexibility and choice and lots of configuration files all over the place.

http://www.youtube.com/watch?v=PQbuyKUaKFo

Don’t Repeat Yourself

- This is a basic tenet of the Object Oriented approach
- You should not have to cut and paste the same code over and over in many places - this leads to future bugs where code is changed one place and not the other.
- Powerful libraries such as Ajax support, and Object Relational Modeling are built in and easy to use with a minimum number of lines of code

Agile Development

- Build quickly, test, evaluate, and refine - iterate from a prototype to production
- You can work independently on application look and feel, storage patterns, and program logic
- Quick development - edit a file - press refresh on a browser
- Includes convenient testing framework

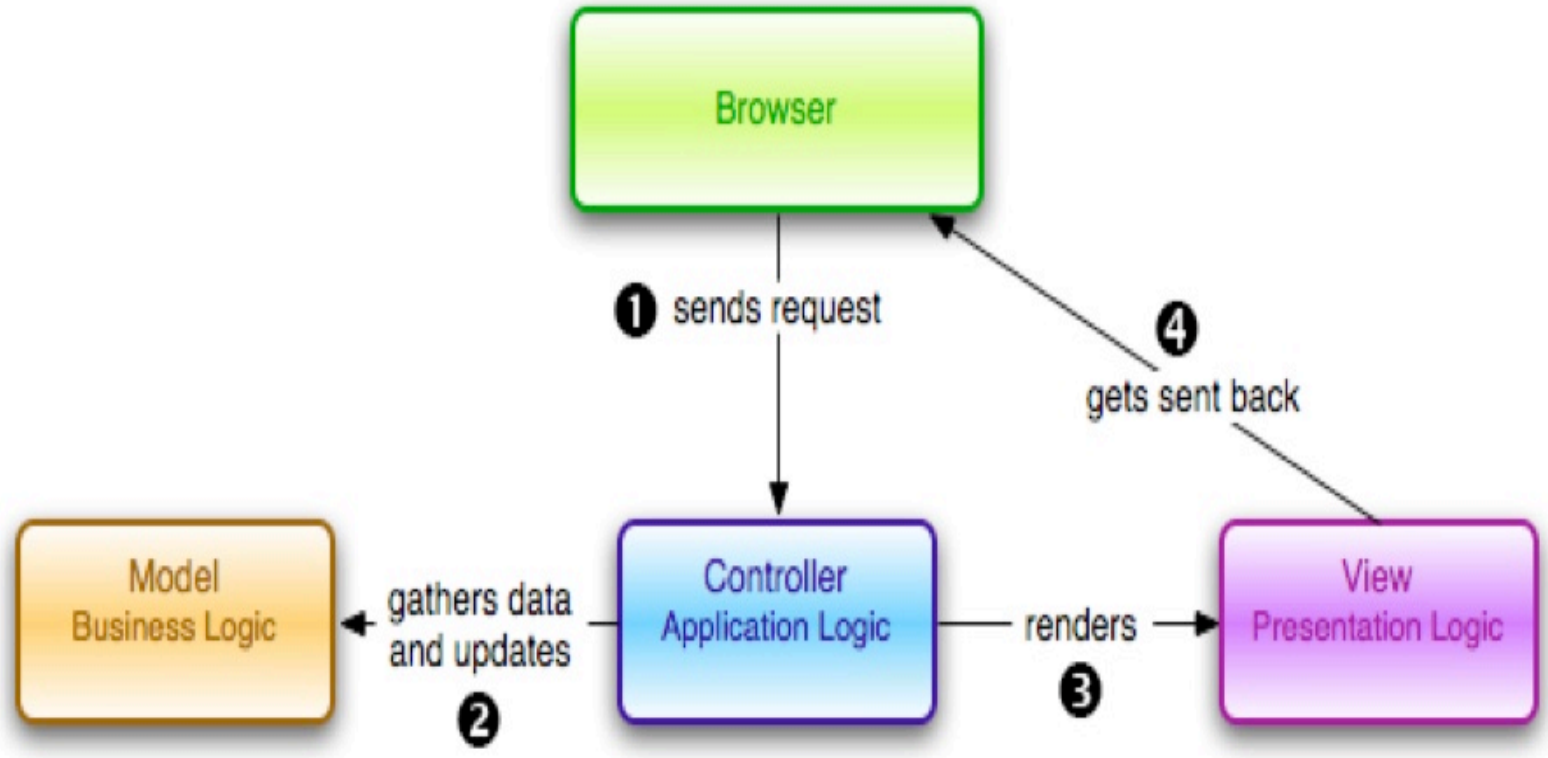
Scaffolding

- Rails provides the framework and will write skeleton applications to use the framework at your direction
- Saves a lot of typing and reading
  - ruby script/generate controller Music index details edit
- Code produced is functional - and has hints about what to do next
- A leg up for beginning programmers - not just a blank slate

Model - View - Controller

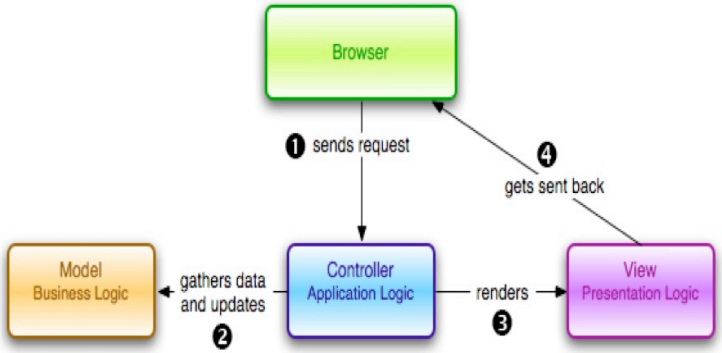
- Model - Persistent Storage - Database - stores across all sessions and across time
- View - Look and Feel of the application - usability, accessibility, design, appeal, functionality, drag and drop, ...
- Controller - Program logic, business rules, flow form screen to screen controls each session independently.

MVC - Request - Response Cycle



MVC Sequence

- User presses button, browser sends data to application
- Controller receives the data, and makes updates to and/or retrieves from the model as necessary
- User output data is passed to the View - view applies final look and feel and the response goes back to the Brower.



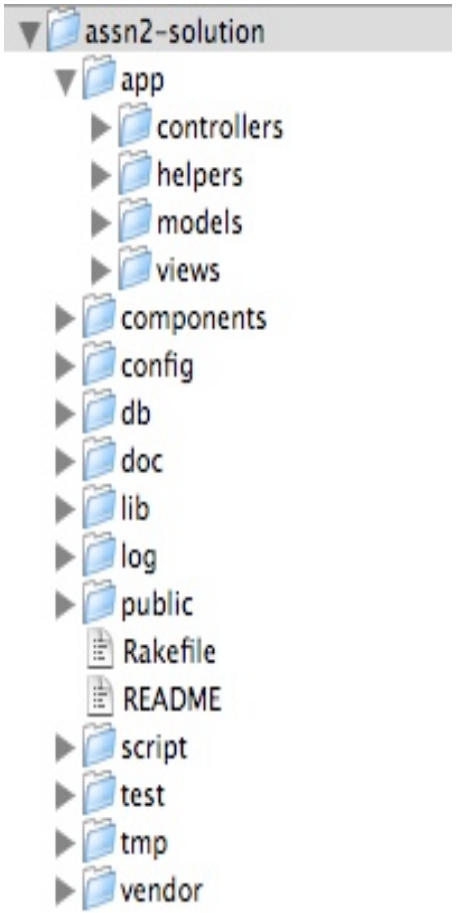
Rails and MVC

- Rails is a MVC application framework - you should just accept the MVC pattern and work with it
- This is OK because MVC is a very popular approach to developing web applications
- Rails effectively pre-chooses 90% of the architecture choices of a web application.

http://www.youtube.com/watch?v=Ld919IziKgE

A Rails Application

- Rails even dictates the layout of an application directory - one less decision for a developer to make
- The directory structure precisely reflects the MVC architecture
- This helps Rails developers know where to look for things when faced with a new Rails application
- Removing choice improves clarity



Potential Rails Gotcha's

- There is no such thing as a perfect framework that does it all for you magically.
- Sometimes you must step around or outside the framework to get things done that the framework is simply not ready to handle
- As we gain experience with Rails we will find areas that need improvement
- The hope if that Rails will need fewer of these "bypass" operations and when it must be done - it can be done elegantly.

Ruby's Future

- Most see Python and Ruby as the likely "what's next?"
- Interesting question - Will Ruby replace perl as the data-grocking language?
- Ruby has been adopted by Sun and Java - Java version 6 will have support for a Ruby in the Java Run Time environment - this will make it much easier for Ruby to become an enterprise language - Ruby will be easy - Java.
- Apple in the 10.5 release makes Ruby as one of the languages which can be used to develop desktop applications in Cocoa - up to now Desktop applications on Macintosh were generally written in a very OO language called Objective C

Rails Future

- Rails is the most innovative full-stack web development framework ever.
  - It is far easier than Java / Spring / Hibernate
  - It is much cleaner than PHP
- For small to medium sized applications starting now with 2-5 year life spans - Ruby is the current smart answer
- For the largest applications with 10+ year life spans, the jury will be out for many years

Conclusion

- Ruby is a very strong contender along with Python as the ideal Object Oriented Language in the market today
- Ruby and Rails are experiencing strong growth at this point in time
- Ruby on Rails is already demonstrated to be well suited for small and medium sized web applications
- More time and experience is needed for Ruby to begin to take Enterprise market share from Java and C# (.Net)